APPLICATION TO WITHDRAW AS ATTORNEY OF RECORD

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Petrocy

Serial No.: 08/807,567 Group Art Unit: 2309

Filed: 02/28/97 Examiner: (not assigned)

For: Self-Addressing Control Units and Modular Sign Including Plurality of Self-Addressing Control Units

---

APPLICATION TO WITHDRAW AS ATTORNEYS OF RECORD

Commissioner of Patents and Trademarks
Washington, D.C. 20231

Sir:

I, the undersigned attorney of record in the above-captioned patent application, hereby apply to the Commissioner to withdraw as the attorney of record in this application for non-payment of legal fees, as set forth in detail below.

1. Between February 18, 1997 and May 8, 1998, we have rendered substantial legal services including, preparation and filing of the patent application; preparation and filing a response of Notice to File Missing Parts; and preparation and filing of Information Disclosure Statement, with copies of references in connection with Applicant's invention pertaining to Self-Addressing Control Units and Modular Sign Including Plurality of Self-Addressing Control Units, as claimed in the present application.

2. In exchange for the legal work, the Applicant agreed to pay us for our services rendered on its behalf.

3. Between March 21, 1997 and June 3, 1998, we submitted invoices to the Applicant for the legal work and the disbursements incurred in connection therewith.

4. To date, the Applicant has not paid us for all of the legal work we provided, which Applicant promised to pay and has only partially paid us for our services and disbursements. Applicant presently owes a substantial balance for the legal work. In addition we have sent numerous letters to Applicant attempting to collect the outstanding balance, and informing the Applicant that we will withdraw as its attorney if we are not paid.

5. Applicant has made numerous promises to us that further payments will be forthcoming. For example, see our letter dated January 23, 1998 (Appendix A) confirming our telephone conversation wherein he indicated that a number of jobs were finishing up and that he would pay us shortly and our letter dated May 8, 1998 (Appendix B) confirming the receipt of a check and indicating that we need payment by June 1, 1998. However, he has not payed his outstanding balance due to us.

6. We believe that we will be harmed if we were to continue to assume responsibility in any way over this application. Specifically, if this application for withdrawal is not approved, then our professional and ethical obligations as attorneys would force us to expend additional time and resources on behalf of the Applicant in monitoring the status of this application and responding to any and all office actions as they arose. However, we would most likely receive no compensation for any of these efforts.

7. The status of this application is that an Office Action dated June 17, 1998 was sent by the Patent Office. A copy was forwarded to Applicant. A Response is due by September 17, 1998, though extensions of time can be obtained to December 17, 1998. Thus, our withdrawal at this time will not prejudice Applicants position and will afford Applicant sufficient time to seek substitute counsel on a timely basis, and to respond to the Restriction Requirement.

8. Applicant has copies of all documents filed with the Patent Office. Such documents are routinely sent to clients as filed.

9. A copy of this Request for Withdrawal is being sent to the Applicant by the letter attached hereto as Exhibit C.

10. Once the application for our withdrawal is approved, kindly direct all future correspondence regarding this application to the Applicant at his address: 24 Orchard Street, Carteret, New Jersey 07008, and direct all telephone calls to Applicant at his telephone number: 732-969-1484.

In conclusion, in the absence of receiving all of our fees from the Applicant, we do not want to expend any further time on this application or continue to be responsible in any way therefor. For this reason, we earnestly solicit the Commissioner's prompt consideration and approval of this application to withdraw.

Respectfully submitted,

Dated: 6/22/98

Michael R. Friscia
Reg. No. 33,884
Attorney for Applicant
FRISCIA & NUSSBAUM
405 Murray Hill Parkway
E. Rutherford, N.J. 07073
Tel. (201) 842-0800
Fax. (201) 842-0229

p/0415301.277

Appendix A

FRISCIA & NUSSBAUM

A Professional Corporation
Attorneys at Law
405 Murray Hill Parkway
East Rutherford, New Jersey 07073
(201) 842-0800

Patents, Trademarks
and Copyrights

Fax (201) 842-0229

New York address:
42-40 Bell Blvd.
Suite 301
Bayside, NY 11361
(718) 224-5080

January 23, 1998

Mr. Richard Joel Petrocy
24 Orchard Street
Carteret, NJ 07008

Re: Our file: 277 - Billing

Dear Joel:

This is to confirm our telephone conversation wherein you indicated that a number of your jobs were finishing up and that you should have money to pay us shortly.

We note that you have previously provided us with a similar update beginning about June 1997. Additionally, you have told us this a number of times thereafter including December 1, 1997, when you paid $1,000.00, and indicated that you would be paying off your balance shortly thereafter. Almost two months later, we still have not received any additional payment.

This is to inform you that if we do not receive payment by February 28, 1998, which is approximately one (1) year after we performed a majority of services, at which time your balance will have been outstanding for approximately one (1) year, we will have no choice but to withdraw as your attorney in your pending patent applications and we will consider our collections options.

We hope to hear from you soon in a positive manner.

Regards.

Very truly yours,

Michael R. Friscia

jr
0123bil.277

Appendix A (7 pages)

```
; START OF PROGRAM DATASIGN EXPERIMENTAL CODE
; FOR USE BY DATASIGN .
; based on serdata4.src for use with arrow message pointer default=1 9/13/94
bit_K       =       24          ;Change this value for desired baudrate is 19.2KBaud for 8 Mhz,9600 Baud
for 4 Mhz
half_bit    =       bit_K/2     ;as shown in table.
;
TOP1        EQU  RA.0; TOP FIRST BIT
BOT         EQU  RA.1; BOTTOM BIT
optoset     equ  ra.3;set data pulse normally high
serial_in   equ  rb.1
direction   equ  rb.6; output
on_off      equ  rb.5; output
Data_clear =  rb.4; output change for pic1654j.pcb artwork
reset_in    =  rb.2
optopwr     =  rb.0
OPTO        EQU  RA;   REFERS TO ALL 4 PINS AS inputs ra.2 & ra.3 tied HI
BRAKE       EQU  rb.7; use motor chip BRAKE input for quicker stops
Shoneytape EQU  rb.3; shoney tape=1, else honey tape
;
            org    8

delay_cntr      ds     1        ;Counter for serial delay routines
bit_cntr        ds     1        ;Number of received bits
rcv_byte        ds     1        ;The received byte
rcv_done   ds 1; its done
Count0     ds  1;Register labels
Count1     ds  1
Number     ds  1
RcvReg     ds  1
DlyCnt     ds  1
; datasign start
DEFAULT    ds 1
lastletter ds 1
newdata    ds 1
Count      ds 1
Datain     ds 1
optostop   ds 1
;
;Flags
FLAG        EQU  1AH.0
lastdirection  EQU  1AH.1
botFLAG     equ  1Ah.2
TOP1FLAG    EQU  1AH.3
FLAG2       EQU  1AH.4;
dataflag    equ  1Ah.5
jumpflag    equ  1Ah.6
R_DONE      equ  1AH.7
; datasign end
; Org 0 sets ROM origin to beginning for program.
```

Gen RAM

→ establish start address

- set aside space for variables

'=' = EQU

ORG =

DS = variable space

5/4

Pep

1/4"

```
                org     0
;
; Set the device type, oscillator type, watchdog timer status, and code
; protect status
;
        DEVICE  PIC16C54,XT_OSC,WDT_OFF,PROTECT_OFF
;
        RESET   Start ;Set reset vector to address at Start
                ;(PIC will jump to this when reset)
;
Start   clrB Flag
        clrb flag2
;               76543210 bit registers
        mov !RA,#00001111b ;Set data direction register for port A 4 INPUTS
        mov !RB,#00001110b;Set data direction register for port B 6/28/94, 1,2,3 input

        clrb lastdirection
        CLRB BOTFLAG
        CLRB TOP1FLAG
        CLRB R_DONE
        clrb dataflag
        setB BRAKE
; new code 9/12/94
        jb optoset,notarrow
        Mov optostop,#4; ra.2 tied HI all else 0
        MOV DEFAULT,#5;blank character Arrow tape
        jmp resume
notarrow  Mov optostop,#12; ra.2 & ra.3 tied HI all else 0
        jb Shoneytape,shoneyyes
        MOV DEFAULT,#23;blank character HONEY tape normally 22 see Joel
        jmp resume
shoneyyes  MOV DEFAULT,#17;blank character SHONEY tape normally 16 see Joel
; end new code
resume
;
        clrb optopwr; turn on FET for optoLEDs
;
        clrf Count
        clrf Count0
        clrf rcv_byte
        setb rcv_done
        clrb Data_clear
;

        mov lastletter,#60;
        MOV NUMBER,#0

        call end_delay; wait a bit
;
        clrB on_off; turns motor on
        setB direction; go to beginning { rewind }
```

```
          cje opto,optostop,tailsafe;6/29/94 change VK 9/13/94
;initialize loop
INITIALIZE JNB FLAG2,BOTLOOP; debounce routine for opto tops and bottom
          CSE OPTO, optostop;both tops and bottom goto 0 at start of tape
          JMP INITIALIZE
;
failsafe   clrb direction; got to start now go forward
Digit      JNB FLAG,UPLOOP
;
          JB TOP1,DIGIT
;
FORWARD    INC NUMBER
          cjne Number,default,Clear
          mov NEWDATA,Number ;save Number
          setb flag2 ;
          jmp center
Clear      clrb flag
          jmp Digit
;
; ***********************************************
; delays
; This delay loop takes four instruction cycles per loop, plus eight
; instruction cycles for other operations (call, mov, the final djnz, and ret).
; These extra cycles become significant at higher baud rates. The values for
; bit_K in the table take the time required for additional instructions into
; account.
bit_delay      mov     delay_cntr,#bit_K
:loop          nop
                 djnz    delay_cntr, :loop
                 ret
;
; This delay loop is identical to bit_delay above, but provides half the delay
; time.
;
start_delay     mov     delay_cntr,#half_bit
:loop          nop
                 djnz    delay_cntr, :loop
                 ret
;
; This delay loop is identical to bit_delay above, but provides long delay
; time.
;
end_delay       mov     delay_cntr,#255
:loop          nop
                 djnz    delay_cntr, :loop
                 ret
;
; ***********************************************
;centering routine used by all
center     mov lastletter,newdata ;save Number
           mov number,#0
```

```
            clrb dataFLAG
            movb jumpflag,direction;move direction bit to jumpflag
;           jb flag2,centered 8/15/94 vk
;original 9/2/94          jb jumpflag,centered;going in reverse I can stop NOW
            jb jumpflag,backflip;going in reverse make sure I am a 0
            mov w, #200  ; 4/1/93 first line
home2        movwf   DlyCnt ;4/1/93 pullback routine to get 0 from top opto
:redo_1     decfsz  DlyCnt,1 ; when going forward
            goto    :redo_1 ; Normally without these lines it would stop
            sb top1; 4/1/93 at a 1 which would screw up next move
            jmp home2;
            setb direction
            mov w, #200
home3        movwf   DlyCnt
:redo_2     decfsz  DlyCnt,1
            goto    :redo_2
            snb top1;
            jmp home3; last 4/1/93 test line
            jmp centered
backflip
            mov w, #200  ; 4/1/93 first line
:home2       movwf   DlyCnt ;4/1/93 pullback routine to get 0 from top opto
:redo_1     decfsz  DlyCnt,1 ; when going forward
            goto    :redo_1 ; Normally without these lines it would stop
            sb top1; 4/1/93 at a 1 which would screw up next move
            jmp : home2;
            clrb direction
            mov w, #200
:home3       movwf   DlyCnt
:redo_2     decfsz  DlyCnt,1
            goto    :redo_2
            snb top1;
            jmp :home3; last 4/1/93 test line


;serdata3 original
centered    clrb BRAKE; use motor chip BRAKE input
            clrb flag2
            setb ON_OFF
            CALL end_delay
            setb optopwr; turn off optoLEDs
;end original serdata3
; ***********************************************
samedigit  nop
;
newdigit
;*********************************************************************
; start serial receive routine
Talk
begin       clrf Count
start_bit   snb  serial_in        ;Detect start bit. Change to
```

```
        jmp  start_bit      ;No start bit yet? Keep watching.
        call start_delay    ;Wait one-half bit time to the middle of the start bit.
;
        jb Serial_in,start_bit
;
;If the start bit is good, proceed. Otherwise, continue waiting.
;
        mov bit_cntr, #8    ;Set the counter to receive 8 data bits
        clr rcv_byte        ;Clear the receive byte for new data.

:receive  call bit_delay    ;Wait one bit time.

        movb c,Serial_in    ;Put the data bit into carry.
        rr   rcv_byte       ;Rotate the carry bit into the receive byte.
;Get next bit
        djnz bit_cntr,:receive
        call bit_delay      ;Wait for stop bit.
;
Displ   mov  newdata, rcv_byte
        setb Data_clear
        clrb rcv_done
        goto wait_bit       ;wait for reset bit after all digits

wait_bit  snb  reset_in     ;Detect reset bit.

        jmp  wait_bit       ;No reset bit yet? Keep watching.
        call start_delay    ;Wait one-half bit time to the middle of the start bit.
        call start_delay    ;Wait one-half bit time to the middle of the start bit.
        jb reset_in,wait_bit

        clrb Data_clear
        setb rcv_done
        JMP SHOWDIGIT
;
bad_digit  jmp begin
;
; end serial receive routine
; ***********************************************
SHOWDIGIT  cje newdata,#80,start;if module is lost it is forced to initialize at start
;
        cje lastletter,newdata,samedigit;if new digit is same as old digit
;                                   ignore it and wait for another
        mov number,lastletter
        clrb optopwr; turn on FET for optoLEDs
        call end_delay; give opto time to come up
        cja newdata,lastletter,goforward;go forward if new digit is greater
        cjb newdata,lastletter,gobackward;go backward if new digit is less
;
; ***********************************************
goforward  clrb direction;set forward direction
        movb jumpflag, direction
```

```
          clrb lastdirection; set lastdirection to 0 for forward
          setb BRAKE;remove brake
          clrb on_off;start motor
upDigit   JNB FLAG,waitlOOP;debounce up
;
          JB TOP1,upDIGIT
:FORWARD  cjne Number,newdata,:Clear
          jmp center
:Clear    INC NUMBER
          clrb flag
          jmp upDigit
;
; ***********************************************
gobackward setb direction;set reverse direction
          movb jumpflag, direction
          setb lastdirection; set lastdirection to 1 for backward
          setb BRAKE;remove brake
          clrb on_off;start motor
downDigit JNB FLAG,waitLOOP;debounce down
;
          JB TOP1,downDIGIT
;
:reverse  cjne Number,newdata,:Clear
          jmp center
:Clear    dec NUMBER
          clrb flag
          jmp downDigit
;
; ***********************************************
; delay and debounce loops
UPLOOP     CLR COUNT0
          MOV COUNT1,#1
:LOOP      JNB TOP1,DIGIT
          DJNZ COUNT0,:LOOP
          DJNZ COUNT1,:LOOP
          SETB FLAG
          JMP DIGIT
;
botLOOP    clr COUNT0
          mov COUNT1,#100; improve debouncing ??
:LOOP      JNB bot,INITIALIZE
          DJNZ COUNT0,:LOOP
          DJNZ COUNT1,:LOOP
          clr COUNT0
          MOV COUNT1,#100
:loop3    JNB TOP1,INITIALIZE
          DJNZ COUNT0,:LOOP3
          DJNZ COUNT1,:LOOP3
          SETB FLAG2
          JMP INITIALIZE
;
```

```
;waitloop for both fwd and rev
waitLOOP   mov COUNT0,#2
           MOV COUNT1,#2
:LOOP      jb jumpflag,:down
           JNB TOP1,upDIGIT
           jmp :goon
:down      JNB TOP1,downDIGIT
:goon      DJNZ COUNT0,:LOOP
           DJNZ COUNT1,:LOOP
           SETB FLAG
           jb jumpflag,:down2
           Jmp upDIGIT
:down2     JMP downDIGIT
;END OF SRC
```

FRISCIA & NUSSBAUM

A Professional Corporation

Patents, Trademarks
and Copyrights

Attorneys at Law
405 Murray Hill Parkway
East Rutherford, New Jersey 07073
(201) 842-0800

Fax (201) 842-0229

New York address:
42-40 Bell Blvd.
Suite 301
Bayside, NY 11361
(718) 224-5080

B

May 8, 1998

Mr. Richard Joel Petrocy
24 Orchard Street
Carteret, NJ 07008

Re: Our files: 277301
SELF-ADDRESSING CONTROL UNITS AND MODULAR SIGN INCLUDING PLURALITY OF SELF-ADDRESSING CONTROL UNITS

277302
MODULAR SIGN BOX WITH FRAME

Dear Joel:

Thank you for your check in the amount of $500.00. As your balance is substantial and has been outstanding for more than one year, we can not change our plans to withdraw as your attorney. We will not file any such papers until **June 1, 1998.** If we do not receive further payment from you by this date, we will proceed with filing a withdrawal.

Regards.

Very truly yours,

Michael R. Friscia

enc.

CC: Billing File

db
b\0507302.277

APP. B

(4 pages)

```
' constants
addro   con     8       WR
cmndo   con     11      WR
cmndi   con     12      Read
baud    con     396
gmove   con     $f8
lreset  con     $f9
last    con     30

' vars
al      var     byte
ah      var     byte
digit   var     byte
stat    var     byte
temph   var     byte
templ   var     byte
tempd   var     byte
temps   var     byte
a       var     byte
nummod  var     byte
b       var     byte
rt      var     byte

' init stuff.
high cmndo
low addro
input cmndi

begin:

' wait for all modules to power on
debug "waiting for modules to power on",CR
pause 7000

'reset all modules first
gosub reset_all_modules

'init
'address modules, then find last one
stat = 0
al = 1
ah = 0
gosub send_address

' now address modules one at a time to see end. note
' max of 100
digit = 0

for a = 1 to last
        al = a
        gosub send_data_nc
        if stat = 1 then cex
next

cex:
' if a = 1, then no modules
if a > 1 then main

debug BELL,"no modules have responded!",CR
```

```
end

' main routine
main:
a = a - 1
debug "found ",SDEC(a)," module(s).",CR


' show the address
for a = 1 to 8
        ' readdress modules, just in case
        al = 1
        gosub send_address

        lookup a,[1,1,2,3,4,5,6,7,8,9],digit
        gosub send_data

        for b = 1 to 8

                al = 2
                lookup b,[1,1,2,3,4,5,6,7,8,9],digit
                gosub send_data

                gosub global_move
        next
next


goto main

ender:

debug "done.",CR
thatsall:
debug BELL
goto thatsall

'''''''''''''''''''''''''''''''''''''''''''''''''''''
local_reset:

debug "lreset al=", SDEC(al), " ah=", SDEC(ah),CR
serout cmndo,baud,10,[ah+$80,al,$f9]
pause 7000

return

'''''''''''''''''''''''''''''''''''''''''''''''''''''
send_address:

debug "addr al=", SDEC(al), " ah=", SDEC(ah),CR

' this line changes the address
' data is sent out AMSB,ALSB
serout addro,baud+$4000,5,[ah,al]
pause 2500      ' should be enough time to address 200 modules

return

'''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
send_data_nc:

debug "data nc dg=", SDEC(digit), " al=", SDEC(al), " ah=", SDEC(ah),CR

' sends data to module, without verify
' data is sent AMSB,ALSB,DIGIT
' if digit = $85, then LOCAL MODULE RESET
serout cmndo,baud,10,[ah+$80,al,digit]
' get responce from module
serin cmndi,baud,1000,nr,[temph,templ,tempd,temps]

debug "got responce",CR
stat = 0
return

nr:
debug "no responce",CR
stat = 1

return
''''''''''''''''''''''''''''''''''''''''''''''''''''
send_data:
stat = 0

for rt= 1 to 3
      debug "data dg=", SDEC(digit), " al=", SDEC(al), " ah=", SDEC(ah),CR


      ' sends data to module
      ' data is sent AMSB,ALSB,DIGIT
      ' if digit = $85, then LOCAL MODULE RESET
      serout cmndo,baud,10,[ah+$80,al,digit]
      ' get responce from module
      serin cmndi,baud,1000,rerror,[temph,templ,tempd,temps]
      debug "verifing responce...",CR

      temph = temph & $7f
      if temph <> ah then rerror
      if templ <> al then rerror
      if tempd <> digit then rerror
      if temps <> 0 then rerror
      goto senddone

rerror:
debug "receive error"
pause 1000
next

debug BELL,"No responce from module",CR
stat = stat + $80
return

senddone:
debug "status =",SDEC(temps),CR
stat = temps
return

'rerror:
'debug BELL,"Receive error.",CR
'debug "    ah-",hex temph,CR
```

```
'debug "   al-",hex templ,
'debug "digit-",hex tempd,CR
'debug " stat-",hex temps,CR
'stat = stat + $40
'return

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
global_move:

debug "gmove",CR

' move to new digit
' global move command
serout cmndo,baud,0,[gmove]
pause 5000
return

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
reset_all_modules

debug "reset all",CR

al = 0
ah = 0
gosub send_address
for al = 0 to last
  serout cmndo,baud,10,[ah+$80,al,$f9]
next
pause 7000
return
```

FRISCIA & NUSSBAUM

A Professional Corporation

Patents, Trademarks,
and Copyrights

Attorneys at Law
405 Murray Hill Parkway
East Rutherford, New Jersey 07073
(201) 842-0800

Fax (201) 842-0229

New York address:
42-40 Bell Blvd.
Suite 301
Bayside, NY 11361
(718) 224-5080

June 22, 1998

Mr. Richard Joel Petrocy
24 Orchard Street
Carteret, NJ 07008

Re: Our file: 277301
Serial No. 08/807,567
SELF-ADDRESSING CONTROL UNITS AND MODULAR SIGN INCLUDING PLURALITY OF SELF-ADDRESSING CONTROL UNITS

Dear Joel:

Enclosed please find a copy of the Application to Withdraw as Attorneys of Record, as filed with the United States Patent and Trademark Office on today's date for the above-referenced matter. This means that we will no longer represent you in this matter. We recommend that you obtain a patent attorney to represent you.

Please be advised that a response to the outstanding Office Action is due by **September 17, 1998** with extensions available by payment of fees until December 17, 1998.

Regards.

Very truly yours,

Michael R. Friscia

enc.

db
06223012.277

PHASE3.SRC

APP. C

(12 pages)

```
        DEVICE  PIC16C84,XT_OSC,WDT_OFF,PROTECT_ON

; timelog
; 1/9   3 hours goto everything together. killed lower opto
; 1/10  2 hours kill extra parts. cleaned up code somewhat
; 1/11  2 hours re-wrote filter. added motor turnoff. wrote new
;               default finder after home.
; 1/12  5 hours  well, re-wrote everything else that vk had
;               added new comm schemes. new centering. parity check
;               started.
; 1/14  2 hour  added ee routines and addressing logic. stated new schematic
; 1/18   5 hour finished schematic, started layout.
; 1/20   2 hours  PCB layout
; Paid... 1200.00
; 1/31   2 hours          pcb assemble 7:00 - 9:00
; 1/31  3 hours           9:00 - 12:00am
; 2/1             12:00am -

; comm at 1200 baud
bit_K   equ     206
half_bit        equ     bit_K/2 ;as shown in table.
DEFAULT equ     3
GCOMMAND        equ     0f5h
ADDRH   equ     0
ADDRL   equ     1
CURDIGIT        equ     2
RT      equ     4       ;number of retries


; port a defs
nc1     equ     0       ;out
comnd_out equ   1       ;command echo back (out - 0)
addr_out        equ     2       ;address out (out - 0)
nc2     equ     3       ;out

; port b defs
optodig         equ     0       ;digit opto input (in)
comnd_in equ    1       ;global command (in)
addr_in equ     2       ;address in (in)
nc3     equ     3       ; out
in2_4   equ     4       ;motor direction (out - 1)
on_offdig equ   5       ;color motor on/off (out - 0)
on_offcol equ   6       ;digit motor on/off (out - 0)
in1_3   equ     7       ;motor direction (out - 1)

;
; data memory
        org     0ch

bit_cntr ds             1               ;Number of received bits
rcv_byte ds             1               ;The received byte
number   ds     1
lastletter ds   1       ;last digit shown (or the current digit shown)
newdata  ds     1       ;used by showdigit
black   ds      1       ;used by getstate
white   ds      1       ;used by getstate
highch  ds      1       ;used by getstate
highcl  ds      1       ;used by getstae
temp    ds      1
tol     ds      1       ;low order to for motors
```

```
        tris    ra
        movlw   00000111b
        tris    rb

        movlw   01010111b       ;set prescale to tmr0, turn on rbres
        option
        bsf     intcon,5                ;enable timer ints

;
; set the module address
        movlw   ADDRH
        movwf   eeadr
        call    read_ee
        movf    eedata,0
        movwf   addressh

        movlw   ADDRL
        movwf   eeadr
        call    read_ee
        movf    eedata,0
        movwf   addressl

;
; see if virgin module

        incf    addressh,0      ; inc and leave in w
        btfss   status,2
        goto    dohome

        incf    addressl,0
        btfss   status,2
        goto    dohome

; a virgin, lets do it
        bsf     flags,virgin    ;indicate a virgin

defaulthome
        movlw   DEFAULT
        movwf   eedata
        movlw   CURDIGIT
        movwf   eeadr
        call    write_ee

dohome
        movlw   CURDIGIT
        movwf   eeadr
        call    read_ee
        movf    eedata,0
        movwf   newdata

tryagain
        movlw   RT
        movwf   retries

homeagain
        call    home
        btfsc   flags,timeout
        goto    herror
        btfss   flags,parity
        goto    newshow
```

```
;
; opps, error homin' twice. if failure, we are a dead module!
; but still allow address data to pass...
herror
        bcf     flags,timeout
        bcf     flags,parity
        decfsz  retries
        goto    homeagain
        call    motor_off
        bsf     flags,deadmod
        goto    waitloop

; show blank char
newshow
        call    showdigit
        btfsc   flags,timeout
        goto    herror
        btfsc   flags,parity
        goto    herror

        movf    number,0
        movwf   lastletter

; ******************************************************
; main loop
; ******************************************************
waitloop
        call    getcommand
        btfsc   flags,cmndrdy
        goto    gotposcmnd
        call    getaddr
        btfsc   flags,addrrdy
        goto    gotaddr
;
; may want to put some supervisor stuf here... like checking the parity
; and to flags
;
        goto    waitloop

gotposcmnd
; first, if a virgin, ignore everything
        btfsc   flags,virgin
        goto    waitloop

; then, if a deadmod, ingore commands
        btfsc   flags,deadmod
        goto    waitloop

; check for address command
        btfss   rcv_byte,7
        goto    waitloop                ;if this bit not high, then not correct

; ok, address byte... check first if global move
        movf    rcv_byte,0
        movwf   temph
        sublw   0fah
        btfsc   status,2
        goto    globalmove
```

```
; get next address byte
gotc1
        call    getcommand
        btfss   flags,cmndrdy
        goto    gotc1

        movf    rcv_byte,0
        movwf   templ


; get data byte

gotc2
        call    getcommand
        btfss   flags,cmndrdy
        goto    gotc2

        movf    rcv_byte,0
        movwf   tempd

; ok, is it my address
        movf    templ,0         ;get into w
        subwf   addressl,0      ;subtract my address
        btfss   status,2
        goto    waitloop

        movf    temph,0
        andlw   07fh            ;kill upper bit
        subwf   addressh,0
        btfss   status,2
        goto    waitloop

; my address, now get next data byte... it the new digit
cmloop
        movf    tempd,0
        movwf   nextdigit
        goto    waitloop                ;do it again!

gotaddr
        movf    rcv_byte,0      ;get first byte
        movwf   temph           ;temp for 1 data

; get next byte of address
gotaddrl
        call    getaddr
        btfss   flags,addrrdy
        goto    gotaddrl
        movf    rcv_byte,0
        movwf   templ

; check if new address
        movf    templ,0
        subwf   addressl,0
        btfss   status,2
        goto    newaddress

        movf    temph,0
        subwf   addressh,0
        btfss   status,2
        goto    newaddress
```

```
; old address, inc by one, then send it
        incf    templ,1
        btfsc   status,2
        incf    temph,1

        movf    temph,0
        call    sendaddr
        clrf    temp
sd1
        decfsz  temp,1
        goto    sd1
sd3
        decfsz  temp,1
        goto    sd3

        movf    templ,0
        call    sendaddr
        goto    waitloop

newaddress
; write new address
        movf    templ,0
        movwf   addressl
        movwf   eedata
        movlw   ADDRL
        movwf   eeadr
        call    write_ee

        movf    temph,0
        movwf   addressh
        movwf   eedata
        movlw   ADDRH
        movwf   eeadr
        call    write_ee
;
; inc it, and send it along
        incf    templ,1
        btfsc   status,2
        incf    temph,1
; send it
        movf    temph,0
        call    sendaddr
        clrf    temp
sd2
        decfsz  temp,1
        goto    sd2
sd4
        decfsz  temp,1
        goto    sd4

        movf    templ,0
        call    sendaddr

; move display back to the default
        goto    defaulthome
;
;
;
```

```
;
; global move command
globalmove
; write it into ee
        movlw   CURDIGIT
        movwf   eeadr
        movf    nextdigit,0
        movwf   eedata
        call    write_ee

        movf    nextdigit,0
        movwf   newdata
        call    showdigit
        btfsc   flags,parity
        goto    tryagain
        btfsc   flags,timeout
        goto    tryagain
        goto    waitloop

;
; send the addres to next module
; w = data to be sent
senddaddr
        movwf   temp
        movlw   8
        movwf   bit_cntr
        bsf     porta,addr_out
        call    bit_delay
sendit
        rrf     temp,1
        btfss   status,0
        bsf     porta,addr_out
        btfsc   status,0
        bcf     porta,addr_out
        call    bit_delay
        decfsz  bit_cntr
        goto    sendit
        bcf     porta,addr_out
        call    bit_delay
        return

getcommand
        bcf     flags,cmndrdy
        btfsc   portb,comnd_in
        return
        call    start_delay
        btfsc   portb,comnd_in
        return

        movlw   8
        movwf   bit_cntr
        clrf    rcv_byte
comndrec
        call    bit_delay
        btfss   portb,comnd_in
        bcf     status,0
        btfsc   portb,comnd_in
```

```
        bsf     status,0
        rrf     rcv_byte,1
;Get next bit
        decfsz  bit_cntr
        goto    comndrec
        call    bit_delay
;
        bsf     flags,cmndrdy
        call    bit_delay
        return

getaddr
        bcf     flags,addrrdy
        btfsc   portb,addr_in
        return
        call    start_delay
        btfsc   portb,addr_in
        return

        movlw   8
        movwf   bit_cntr
        clrf    rcv_byte
addrrec
        call    bit_delay
        btfss   portb,addr_in
        bcf     status,0
        btfsc   portb,addr_in
        bsf     status,0
        rrf     rcv_byte,1
;Get next bit
        decfsz  bit_cntr
        goto    addrrec
        call    bit_delay
;
        bsf     flags,addrrdy
        call    bit_delay
        return

;
; shows digit in newdata
showdigit
; showing a digit >80 not allowed
        movlw   80
        subwf   newdata,0
        btfsc   status,2
        goto    start           ;if new data greater than 80, restart

; same digit, do nothing
        movf    newdata,0
        subwf   lastletter,0
        btfsc   status,2
        return

; ok, save this digit
        movf    lastletter,0
        movwf   number

; now figure out which way to go...
        movf    lastletter,0
        subwf   newdata,0
```

```
        btfss   status,0
        goto    gobackward

goforward
        call    motor_on_rewind

; wait unit white
foreblk
        call    getstate
        btfss   status,0
        goto    foreblk
foremove
        btfsc   flags,timeout
        return
        call    getblack
        movwf   lastwidth
        incf    number,1
        movf    number,0
        subwf   newdata,0
        btfss   status,2
        goto    foremove

        call    motor_off       ; halt motor
;
; now check for parity
;       movf    lastwidth,0     ;get width in w
;       xorwf   number,0                ;xor with the number
;       movwf   temp
;       btfss   temp,0          ;if temp.0 = 1 then same party
;       goto    forecenter
;       bsf     flags,parity
;       return
;
; now center optodig
forecenter
        call    motor_on_forward        ;start motion
; find black
forw
        call    getstate
        btfsc   status,0
        goto    forw
; find white
for1
        call    getstate
        btfss   status,0
        goto    for1
        call    motor_off

        call    motor_on_rewind ;start motion
; reverse for black again
for2
        call    getstate
        btfsc   status,0
        goto    for2
        call    motor_off

        movf    number,0
        movwf   lastletter

        return
```

```
gobackward
        call    motor_on_forward

; at black, wait until white
backblk
        call    getstate
        btfss   status,0
        goto    backblk
backmove
        btfsc   flags,timeout
        return
        call    getblack
        movwf   lastwidth
        decf    number,1
        movf    number,0
        subwf   newdata,0
        btfss   status,2
        goto    backmove

        call    motor_off       ; halt motor
;
; now check for parity
;       movf    lastwidth,0     ;get width in w
;       xorwf   number,0                ;xor with the number
;       movwf   temp
;       btfss   temp,0          ;if temp.0 = 1 then same party
;       goto    backcenter
;       bsf     flags,parity
;       return
;
; now center optodig
backcenter
        call    motor_on_rewind ;start motion

; ok, find black
back1
        call    getstate
        btfsc   status,0
        goto    back1
        call    motor_off

        movf    number,0
        movwf   lastletter
        return


;
; turns on clears tmr0, to counter, turns on gie, and turns on motor
motor_on_forward
        btfsc   flags,timeout
        return
        clrf    tmr0
        clrf    toh
        clrf    tol
        bcf     portb,on_offdig
        bsf     intcon,7
        bcf     portb,in1_3
        bsf     portb,in2_4
        bsf     portb,on_offdig
        return
```

```
motor_on_rewind
        btfsc   flags,timeout
        return
        clrf    tmr0
        clrf    toh
        clrf    tol
        bcf     portb,on_offdig
        bsf     intcon,7
        bsf     portb,in1_3
        bcf     portb,in2_4
        bsf     portb,on_offdig
        return

motor_off
        bcf     intcon,7
        bsf     portb,in1_3
        bsf     portb,in2_4
        return

;
; filters optodig input. counts black/white, and returns
; which count had more.
; returns  carry = 0 if white, 1 = black
getstate
        movlw   200                     ; 8mhz 100 => 200
        movwf   temp
        clrf    white
        clrf    black
isoptodig
        btfsc   portb,optodig
        goto    iswhite
isblack
        incf    black,1
        goto    getcont
iswhite
        incf    white,1
        nop
getcont
        decfsz  temp
        goto    isoptodig
        movf    white,0
        subwf   black,0
        btfss   status,0
        retlw   0
        retlw   1

;
; get width of black mark
; returns
; 0 = short
; 1 = long
; 2 = end
getblack
        clrf    highch          ;high count
        clrf    highcl

waitblack
        btfsc   flags,timeout
        return
```

```
        call    getstate
        btfsc   3,0
        goto    waitblack

countblack
        btfsc   flags,timeout
        return
        incf    highcl,1
        btfsc   3,2
        incf    highch,1

        btfsc   highch
```

```
toh     ds      1       ;high " "  "  "  "
flags   ds      1       ;general flags
lastwidth ds    1       ;last width of pulse
retries ds      1
addressh        ds      1
addressl        ds      1
commandh        ds      1
commandl        ds      1
nextdigit ds    1
templ   ds      1
temph   ds      1
tempd   ds      1
tdelay  ds      1

;
;flags defs
;
timeout equ     1
parity  equ     2
cmndrdy equ     3
addrrdy equ     4
deadmod equ     5
virgin  equ     6

;
; start of reset jump
        org     0
        goto    start

;
; int routine
        org     4
        incf    tol,1
        bcf     intcon,2
        btfss   tol,6
        retfie
;
; if we get here, we have and error in the motor
motor_error
        bcf     intcon,7
        bcf     portb,on_offdig
        bsf     flags,timeout   ;indicate an error
        retfie

;
; start of main code
start

;
; init stuff
        clrf    status
        clrf    flags
        clrf    retries

        movlw   00000000b
        movwf   porta
        movlw   10010000b
        movwf   portb

        movlw   00000000b
```